UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

INICIACIÓN A LA PROGRAMACIÓN MEDIANTE ROBOTS

Carlos González Sacristán
Tutor: Pilar Rodríguez Marín

INICIACIÓN A LA PROGRAMACIÓN MEDIANTE ROBOTS

Autor: Carlos González Sacristán
Tutor: Pilar Rodríguez Marín

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Carlos González Sacristán
Iniciación a la programación mediante robots

Carlos González Sacristán
C/ Sierra Gádor, 15
Madrid, Madrid 28031

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A Freya y Skadi,
por ser unas ratas del demonio adorables.

"If the automobile had followed the same development cycle as the computer,
a Rolls-Royce would today cost $100, get a million miles per gallon,
and explode once a year, killing everyone inside."

Robert X. Cringely

Agradecimientos

En primer lugar querría agradecer a mis padres el apoyo que me han dado a lo largo de estos años de carrera, apoyándome cuando lo he necesitado, y otras haciéndome salir de mi cuarto a regañadientes, porque también lo necesitaba. A Cristina, que sólo me conoce como estudiante de informática, con todo lo que eso significa, aguantando largas charlas sobre tecnologías que ni le interesaban ni muchas veces entendía.

Quiero agradecer a Pilar, mi tutora del Trabajo de Fin de Grado, la motivación y las buenas ideas, tanto con la memoria como con todo el proyecto, aportando ese punto crítico justo que hace avanzar las cosas por buen camino.

Quiero agradecer a mis compañeros de estudios por todos los buenos momentos, por todas las conversaciones, y porque todos hemos compartido muchas veces ese momento de fatiga y frustración en que lo único que te sale es reírte a carcajadas. Las buenas “horas de las risas”. Gracias a todos por hacer más amenas las noches de prácticas y estudio en mi casa, o en casa de cualquiera, por compartir unas cervezas bien merecidas, y por todos los recuerdos que me llevo de esta carrera.

No puedo dejar de mencionar a dos personas sin las cuales nada de esto habría sido posible: Diego y Lorenzo, que siempre tienen una sonrisa y un café cuando más falta hacen.

Resumen

La iniciación a la programación puede ser una tarea frustrante y difícil para muchas personas debido al cambio drástico que supone cambiar la forma de estructurar los pensamientos y convertirlos en código. En la década de los 60, el MIT comenzó el proyecto LOGO: compuesto de un robot físico y un lenguaje de programación derivado de Lisp, promoviendo una forma diferente de enseñar no sólo programación, sino muchas otras otras asignaturas. El robot ejecutaba las órdenes introducidas en el programa dejando un rastro pintado con rotulador al mismo tiempo que se movía, a voluntad del programador. La iniciación a la programación se convirtió gracias a LOGO en un juego educativo. LOGO fue un éxito, pero tenía, entre sus defectos, el hecho de que el robot físico era caro y complejo, por lo que se sustituyó rápidamente por una versión virtual.

Este Trabajo de Fin de Grado presenta un experimento para evaluar la eficacia de este tipo de plataformas educativas alternativas con las tecnologías actuales. Como herramienta principal para este estudio, se desarrolla *Phogo*, un pequeño robot físico, construido a base de piezas impresas en 3D y componentes electrónicos de bajo coste que se hace uso de tecnologías web estándares para su control. El lenguaje de programación utilizado para controlarlo es Python 3, un lenguaje potente y con muy buenas características como lenguaje de aprendizaje. *Phogo* incorpora un sensor de distancia por ultrasonidos con el que podemos leer datos del mundo real e incorporarlos al código, pudiendo de esta forma interactuar con el entorno.

Palabras clave

robots, programación, LOGO

Abstract

Getting started with programming is a hard and frustrating task for many, due to the drastic change in structuring one's thoughts and turning them into code. In the 60's, MIT began the LOGO project: using a physical robot and a programming language derived from Lisp, they started a new way of teaching, not only programming but other subjects too. The robot executed the commands given by a programmer and and was able to leave a trace of its path at the programmer's will. Thanks to LOGO, programming became an educational game. LOGO was a success but had some drawbacks, among which the fact that the robot was complex and expensive to build, and so it was soon replaced by a virtual version.

This bachelor thesis presents an experiment to assess the capabilities of alternative educational platforms made with today's technologies. It introduces *Phogo*, a small physical robot built with 3D printed parts and low cost electronic components, using standard web technologies to control it as its main tool. The chosen programming language is Python 3, a powerful and well-suited language for teaching. *Phogo* has a built-in ultrasonic distance sensor that allows it to gather data from the real world to be used inside the code, allowing it to interact with its surroundings.

Keywords

robots, programming, LOGO

ÍNDICE

Listas

Lista de figuras

1

Introducción

Hoy en día es prácticamente imposible pasar un día sin depender en cierta forma de un sistema de información, ya sea un ordenador o cualquier otro dispositivo similar: cualquier pregunta que podamos hacernos nos conduce a fuentes de información más o menos relevantes por medio de algoritmos de búsqueda; un algoritmo planifica por nosotros la mejor ruta para ir al trabajo, e incluso las noticias que consumimos son seleccionadas por algoritmos de red social. Estos son sólo algunos ejemplos de cómo el mundo está cambiando por la Revolución Digital [1, 2].

La creciente disponibilidad de tecnología de impresión 3D y el cada vez más bajo coste de los componentes electrónicos están atrayendo cada vez más público al mundo de la computación, la programación y la ingeniería electrónica, como se puede apreciar en el crecimiento del movimiento "maker", dónde cada vez más personas de múltiples ámbitos se unen en una gran comunidad con la capacidad de dar forma a ideas innovadoras. Aunque la barrera de entrada es cada vez más pequeña, siguen siendo necesarios ciertos conocimientos para manejar los tecnicismos relacionados con el mundo de la computación. Esto supone un obstáculo principalmente para jóvenes estudiantes, que podrían beneficiarse de curvas de aprendizaje más graduales.

Todos necesitamos cierta comprensión sobre tecnología, ordenadores y sistemas de información para entender el moderno contexto social en el que vivimos [3]. Por tanto, cualquier educación que tenga como objetivo capacitar al alumno y quiera darle una visión no sólo de cómo usar estas herramientas, sino de cómo funcionan, debe reconocer la realidad digital en que nos encontramos.

Los beneficios de una educación en que la programación tiene cierto protagonismo han sido ampliamente estudiados [4, 5], aunque sigue habiendo dudas en cuanto a si los resultados pueden ser reproducidos en otras áreas, no solo en matemáticas o robótica [6, 7]. En este contexto, ciertos estudios han mostrado que proporcionar a los estudiantes un robot como sujeto de estudio puede ayudar a captar su atención y facilitar su experiencia de aprendizaje [8, 9]. Las plataformas educativas diseñadas para desarrollar este tipo de razonamiento pueden ser un recurso pedagógico valioso.

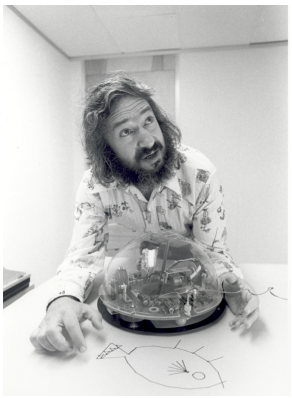


Figura 1.1: Seymour Papert con una *Tortuga*

A finales de la década de los 60, Seymour Papert, Cynthia Solomon y otros investigadores del Laboratorio de Inteligencia Artificial del **MIT!** comenzaron el proyecto LOGO. Diseñaron, desarrollaron y usaron en varios talleres y actividades un sistema compuesto por un robot programable, "*The Turtle*" ("*la Tortuga*"), usando un lenguaje derivado de Lisp, y conocido como LOGO. El robot era capaz de moverse e ir dejando un rastro tras de sí gracias a un rotulador que llevaba incorporado, siguiendo los comandos

implementados por el usuario en el programa. La intención y ambición del proyecto LOGO era revolucionar el sistema educativo y para ello desarrollaron diferentes robots y probaron diferentes enfoques. LOGO fue un éxito, pero los robots eran un recurso caro y complejo para el estado de la tecnología, por lo que rápidamente se sustituyó el robot físico por una simulación, facilitando en gran medida su difusión. LOGO fue retirado del ámbito educativo en la década de los 90 por 3 motivos principales: la complejidad de la sintaxis para algunos estudiantes, la poca importancia de los logros educativos conseguidos (muchas veces limitados a la geometría y patrones visuales), y a la soledad del entorno de aprendizaje [10].

A medida que el precio de los componentes sigue bajando, y gracias a Arduino, la impresión 3D, y otras plataformas y tecnologías, es muy fácil hoy en día prototipar pequeños robots móviles [11]. Por tanto, tenemos la posibilidad de recuperar la idea original de LOGO, con un robot Tortuga físico, a un precio asequible, y usando tecnologías hoy en día comunes y estandarizadas.

En este Trabajo de Fin de Grado se presenta la segunda versión de "*Phogo*", una plataforma educativa basada en un robot de bajo coste impreso en 3D, capaz de dejar un rastro con un rotulador mientras se mueve, así como las herramientas software que permiten una comunicación inalámbrica con el robot, y la interfaz que permite programarlo desde un navegador web.

A continuación se detallará la motivación de este Trabajo de Fin de Grado, seguido de los objetivos principales y una descripción estructural de este documento.

1.1. Motivación

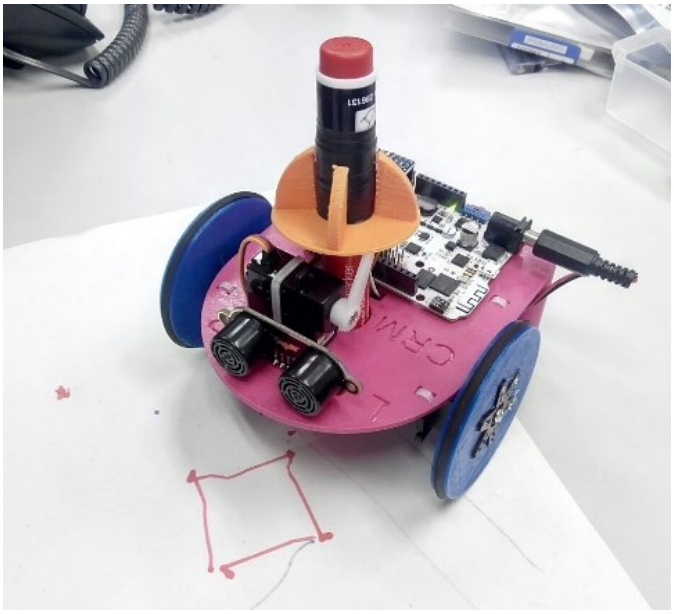
A la hora de iniciarse en el mundo de la programación, no es infrecuente que un alumno se sienta frustrado o agobiado por la falta de resultados visibles o por la dificultad de tener que cambiar de modelo de pensamiento para adaptarse al lenguaje que se está aprendiendo, pero puede reducirse con la motivación y las herramientas adecuadas [12].

Existen muchos lenguajes de programación que podrían usarse para la iniciación a la programación. Por sus características como lenguaje de programación [13], facilidad de uso, expresividad, soporte, comunidad, orientación a la enseñanza y, sobre todo, por ser un lenguaje con el que el alumno puede continuar su aprendizaje más allá del ámbito propuesto por *Phogo*, se ha elegido Python 3 como lenguaje base para la programación del robot.

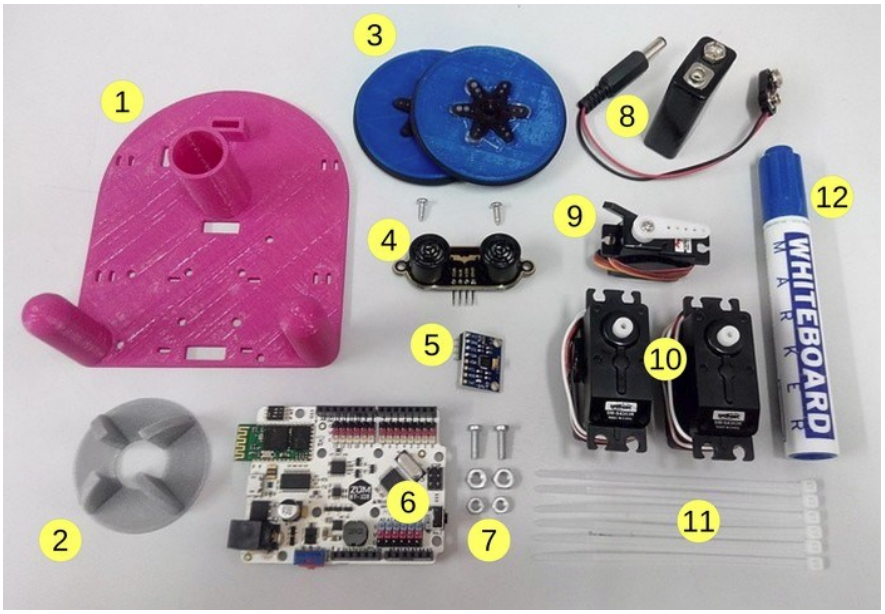
Este Trabajo de Fin de Grado parte de una experiencia previa, llevada a cabo en el CRM-UAM (Club de Robótica y Mecatrónica de la Universidad Autónoma de Madrid), la primera versión de *Phogo*, a principios de verano de 2016, con la misma idea que aquí se desarrolla pero con una implementación del robot y su interfaz de control completamente diferentes [14]. Se ha querido continuar desarrollando el proyecto, aunque volviendo a diseñarlo e implementarlo desde cero, para corregir muchos de los fallos que se vieron en aquél primer intento.

Haciendo homenaje a LOGO, el proyecto anterior se bautizó como "**Phogo**" (en composición libre de "Python" y "LOGO") y al robot físico, *Tortuga* (en inglés, *Tortoise*, a diferencia de LOGO, que llamaba al suyo *Turtle*).[1]

[1] En inglés, ***Turtle*** hace referencia a las tortugas marinas, mientras que ***Tortoise*** hace referencia a las tortugas terrestres.



(a) Una Tortuga completa.



(b) Las Tortugas están hechas con piezas comunes: 1) Chasis imprimible en 3D, 2) Soporte para el rotulador imprimible en 3D, 3) Ruedas imprimibles en 3D, 4) Sensor de distancia, 5) Giroscopio, 6) Placa de control: bq Zum, 7) Tornillos y tuercas, 8) Batería de 9V, 9) Servo para controlar el rotulador, 10) Servos principales, 11) Bridas, 12) Rotulador de pizarra

Figura 1.2: La primera versión de *Phogo*.

1.1.1. Hardware

La Tortuga se diseñó para dar múltiples opciones a los usuarios. Es un pequeño robot con ruedas que se puede mover por una superficie plana, con un sensor de distancia al frente y un rotulador que puede controlarse fácilmente por código, haciendo que baje o suba, entrando en contacto con la superficie por la que se desplaza. Este rotulador permite ir dejando rastro a voluntad del usuario, dando la posibilidad de ir dibujando formas con el desplazamiento del robot.

En esa versión, conseguir precisión en el movimiento fue problemático, puesto que los motores eran servos de rotación continua, que tenían muy poca precisión, y por ello se movían a veces casi independientes el uno del otro: la Tortuga era prácticamente incapaz de moverse en línea recta. El movimiento se terminó calibrando con un giroscopio [15], con lo que se consiguieron líneas más rectas, pero ningún intento de conseguir precisión en la distancia recorrida dio un resultado satisfactorio, al menos sin introducir bastante complejidad en el código de control del robot, lo que se traduce en tiempo de procesamiento. Al menos las Tortugas sí eran más o menos constantes en su desplazamiento, por lo que no se hicieron más intentos de conseguir precisión.

La placa de control elegida fue una bq Zum, principalmente por tener un entorno de programación conocido y por llevar la comunicación Bluetooth integrada. La comunicación con el ordenador de control era totalmente transparente para el usuario, y una vez equipada con una batería de 9V, la Tortuga podía moverse libremente, sin cables.

El diseño era lo bastante simple como para que casi cualquier persona con unos mínimos conocimientos de electrónica, o con capacidad para seguir unas instrucciones, pudiera montarlo. También era un diseño bastante económico, con un coste de unos 80€ por Tortuga.

1.1.2. Software

Todo el código de manejo y comunicación con el robot se encapsuló en un paquete de Python. Para comenzar a usar la Tortuga, no era necesario más que escribir `from phogo import *` en la primera línea del programa para iniciar la conexión inalámbrica con el robot asignado para cada usuario. Una vez establecida la conexión, todas las acciones eran realizadas por la Tortuga asignada.

Las funciones disponibles para el uso de la primera versión de la Tortuga eran similares a las que proporcionaba LOGO:

1.– `forward(units=10)`: La Tortuga avanza el número de unidades especificadas.

2.– `back(units=10)`: La Tortuga retrocede el número de unidades especificadas.

3.– `right(degrees=90)`: La Tortuga gira en sentido horario los grados especificados. El rotulador se encuentra en el centro de rotación, lo que permite dibujar esquinas con precisión.

4.– `left(degrees=90)`: La Tortuga gira en sentido antihorario los grados especificados.

5.– `pen_up()`: Se levanta el rotulador, con lo que deja de pintarse traza bajo el robot.

6.– `pen_down()`: Se baja el rotulador, con lo que comienza a dejarse traza bajo el robot.

7.– `obstacle()`: La Tortuga mide la distancia al obstáculo que tiene enfrente y devuelve el resultado en centímetros.

Además de estos comandos, esta primera versión disponía de una "*Tortuga virtual*". Se podía cambiar el receptor de los comandos entre la Tortuga real y la virtual incluyendo dos instrucciones en el programa: `simulated()` para comenzar a controlar la versión virtual, y `real()` para volver a controlar el robot físico.

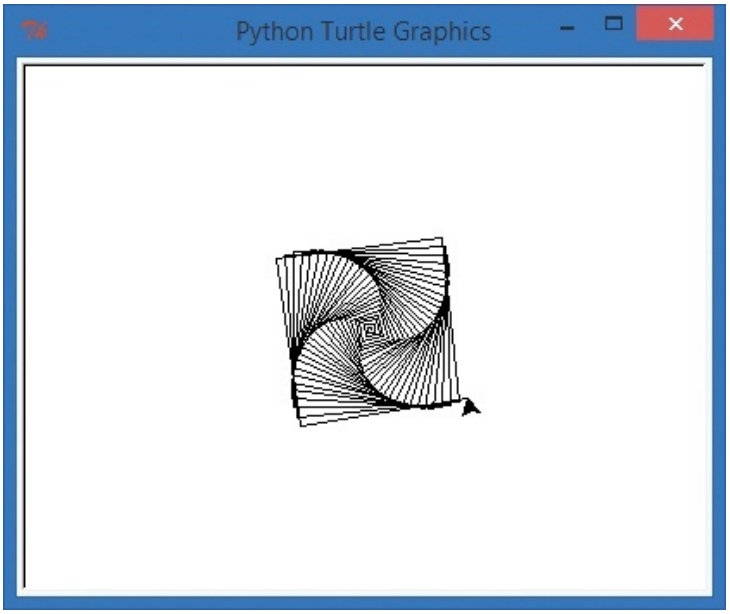


Figura 1.3: La *Tortuga virtual*.

La forma de programar esta versión de *Phogo* era a través de un editor de texto y una terminal, o a través de un editor que permitiera la ejecución de código. Se recomendaba el uso del editor IDLE3, generalmente incluido con las distribuciones de Python 3.

1.1.3. Taller

En junio de 2016, el CRM-UAM, dentro del marco del programa "Campus Inclusivo, Campus sin Límites", llevó a cabo un taller informal de robótica con estudiantes de secundaria en riesgo de no continuar estudios universitarios. Era un grupo mixto de 19 estudiantes de entre 16 y 18 años sin experiencia previa en programación o robótica. Dicho grupo se componía de perfiles muy diversos, pues 12 de ellos tenían algún tipo de diversidad funcional o intelectual: había estudiantes con ceguera, sordomudez, discapacidad motriz y trastorno del espectro autista. Estaban acompañados de sus monitores, un intérprete de signos y 4 voluntarios del CRM-UAM, que guiaban a los estudiantes durante la actividad y contestaban a sus preguntas. Todos los participantes fueron capaces de programar su Tortuga asignada, en solitario o con ayuda. Pueden verse resultados más detallados en el artículo que describe la experiencia [14].

Para la realización de este taller, se construyeron un total de 7 Tortugas. Dada la infraestructura disponible durante el taller, dónde los ordenadores utilizados por los alumnos no tenían posibilidad de conexión Bluetooth, se usaron dos ordenadores "maestros". Cada uno estaba conectado a varias Tortugas: 4 en uno y 3 en el otro. Los ordenadores que usaron los estudiantes estaban, a su vez, conectados a estos ordenadores "maestros". Al finalizar el taller, y viendo los problemas de puesta en marcha que conllevó usar 7 Tortugas con este sistema, se comenzó a pensar en formas de simplificarlo, momento en que nació la idea para este Trabajo de Fin de Grado.

1.2. Objetivos

Este Trabajo de Fin de Grado tiene como objetivo evaluar la eficacia de este tipo de plataformas de aprendizaje alternativas en la iniciación a la programación. Para poder llevar a cabo este objetivo, se necesita un robot programable de bajo coste y versátil, que permita introducir los conceptos básicos de la programación (variables, saltos condicionales, bucles, funciones, ...).

En primer lugar se diseñará e implementará un robot y su interfaz de control. El robot aceptará y ejecutará órdenes en tiempo real del usuario, por medio de un programa Python 3 escrito en la interfaz web. La interfaz de control consistirá en una página web que permitirá programar dicho robot, desde un navegador web, para que realice las acciones solicitadas.

1.2.1. Requisitos del robot

El robot debe ser capaz de dejar un rastro pintado con un rotulador a la vez que se desplaza, al igual que hacía LOGO, pero a diferencia de éste, debe ser inalámbrico. El usuario debe ser capaz de controlar por medio de código si el robot pinta o no.

El robot deberá poder:

- Establecer un punto de acceso WiFi o conectarse a uno ya existente.
- Ejecutar un conjunto de comandos, que se traducirán en acciones a realizar por el robot: avanzar, retroceder, girar...

La interfaz contará con:

- Cuadro de texto o editor para escribir el código que el robot debe ejecutar.
- Cuadro de texto para mostrar al usuario el resultado, enviado por el robot, de los comandos ejecutados.

Las condiciones que se han establecido para la realización del proyecto son las siguientes:

- El robot establece un punto de acceso WiFi o se conecta a uno ya existente.
- El robot hace las veces de servidor web, sirviendo él mismo la interfaz web al navegador del usuario y respondiendo a los comandos enviados desde el cliente.
- El robot ejecuta los comandos que se le piden y devuelve un mensaje con el resultado de la ejecución, que se muestra en la interfaz.
- La interfaz permite al usuario programar el robot en código Python 3 y ejecutar dicho programa.
- La interfaz permite al usuario salvar el código en su propio ordenador y volver a cargarlo si posteriormente así lo quiere.
- La interfaz debe adaptarse a la pantalla del dispositivo que se utilice el usuario.

1.2.2. Planificación del taller

El público objetivo de este taller serán alumnos de los últimos cursos de instituto, sin conocimientos previos de programación, pero con voluntad de aprender. Al ser un taller de iniciación, no debería ser muy largo, ni deberían introducirse conceptos complejos (recursividad, orientación a objetos, ...). El objetivo final es poder evaluar formalmente la viabilidad de este tipo de plataformas como herramientas educativas, y no sean meros juguetes.

Los conceptos que se deberían introducir en este taller, aunque están sujetos a variación, dependiendo de las condiciones en que se lleve a cabo, son los siguientes:

1.– Manejo de las funciones propias de *Phogo*.

2.– Uso de variables.

3.– Saltos condicionales: `if...`

4.– Saltos condicionales: `if...elsif...`

5.– Saltos condicionales: `if...elsif...else...`

6.– Bucles `for`.

7.– Bucles `while`.

8.– Funciones.

1.3. Organización de la memoria

Este documento se compone de las siguientes secciones:

- **Estado del arte**
  En esta sección se revisa la disponibilidad en el mercado de robots programables, con propósito educativo y con capacidad para dejar una traza pintada en la superficie por la que se mueven.
- **Análisis**
  Este apartado detalla los requisitos del sistema.
- **Desarrollo**
  Se detalla en esta sección el proceso de desarrollo del robot, del protocolo de intercambio de datos, y de la interfaz web para el control del robot.
- **Pruebas y evaluación de resultados**
  En esta sección se exponen las pruebas realizadas y los resultados obtenidos durante el desarrollo del robot.
- **Conclusiones, Problemas y Trabajo futuro**
  Aquí se detallan las conclusiones tras la realización del proyecto, los problemas encontrados, los fallos que persisten, y las líneas de trabajo futuro que se desean desarrollar.

2

Estado del arte

El proyecto LOGO sirvió de base para muchos otros proyectos posteriores. Algunos de ellos más conocidos, como Lego Mindstorms, y otros no tanto, o con menos trayectoria. En esta sección se presentan los proyectos más actuales y que más similitud tienen con *Phogo*, es decir, robots programables, concebidos con el objetivo facilitar el aprendizaje o de enseñar a programar y con capacidad para dejar un rastro pintado en la superficie por la que se desplazan. Existen muchos otros robots programables, pero su modo de funcionamiento, en la mayoría de los casos, no es ser programados en un lenguaje de más alto nivel, sino proveer una plataforma para facilitar la programación del robot en sí.

2.1. Proyectos comerciales

2.1.1. Lego Mindstorms

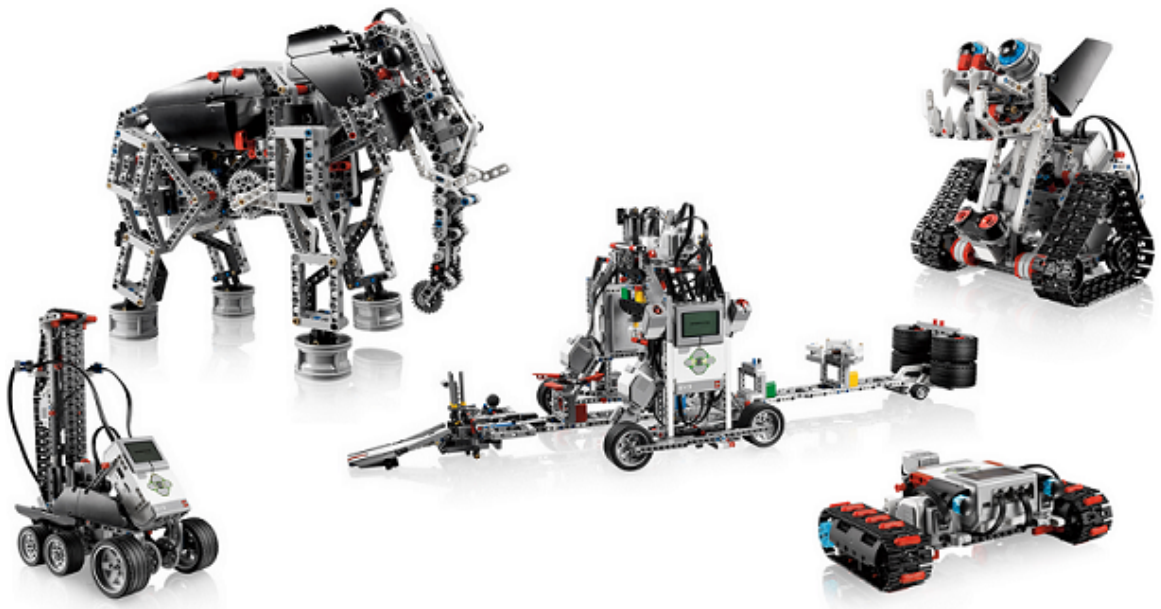


Figura 2.1: Lego Mindstorms EV3.

Lego Mindstorms [16] ha sido el sucesor directo de LOGO. Nació de la colaboración directa entre la compañía de juguetes Lego y el MIT desde el año 1986.

Lego Mindstorms es una línea de juguetes de robótica para niños fabricado por la empresa Lego. Los kits contienen software y hardware para construir robots personalizables y programables. Incluyen un *ladrillo inteligente* que controla el sistema, sensores modulares, y piezas de la línea "Lego Technic" para construir las partes mecánicas. Las raíces de Lego Mindstorms se remontan al "ladrillo programable" creado en MIT Media Lab, que estaba programado en "Brick Logo".

El kit original contenía dos motores, dos sensores táctiles y un sensor de luz. La versión NXT contiene tres servo motores, un sensor de luz, de sonido y de distancia, así como un sensor táctil. Lego Mindstorms puede usarse para modelar un sistema empotrado con partes electromecánicas controladas por ordenador: esto permite modelar muchos sistemas empotrados reales, desde controladores de ascensor hasta robots industriales.

Mindstorms debe su nombre al libro *Mindstorms: Children, Computers, and Powerful Ideas*, escrito por Seymour Papert, 1980.

Hasta la fecha ha habido 3 generaciones de Lego Mindstorms: RCX, NXT y EV3, que data de septiembre de 2014.

2.1.2. Root

Figura 2.2: Root dibujando un fractal.

Root [17] es un producto comercial, financiado por medio de crowdfunding en Kickstarter [18] y fabricado por "Scansorial". La campaña de financiación se completó con éxito a finales de noviembre de 2016, y se esperaban entregas de los primeros robots a los patrocinadores en julio de 2017, pero se ha retrasado dicha entrega, por motivos desconocidos a los no patrocinadores, y la nueva fecha prevista es marzo de 2018.

Root es un robot de tamaño compacto que tiene como objetivo ser una herramienta para enseñar a usuarios, con distintos niveles de conocimiento, a programar. Con un precio durante la campaña de financiación de $195, su característica más llamativa es que puede moverse por una pizarra blanca usando imanes para mantenerse y desplazarse por la superficie vertical. Equipado con un rotulador para pizarras, puede ir pintando en ésta a

medida que se desplaza, o puede ir borrando las líneas. Está concebido para que sea fácil interaccionar con otros Root, para hacer carreras por circuitos pintados en la pizarra e interaccionar de distintas formas con las líneas pintadas, permitiendo varios tipos de juegos. Puede responder también a la intensidad de luz ambiental, presencia de otros Root, e incluso a toques físicos del usuario.

Root tiene 3 niveles de control, para distintos niveles de usuario. El primero, llamado “Graphical Programming” (Programación Gŕafica) pensado para usuarios más jóvenes, es la programación por bloques, usando bloques coloreados como unidad básica. En este nivel, incluso los niños que no saben leer, pueden programar. El segundo es un sistema similar a Scratch, también por bloques, llamado “Computational Fluency” (Fluidez Computacional), que incluye un conjunto de instrucciones más extenso que el anterior. El tercer modo “Full text programming” es la programación con código JavaScript, Python o Swift.

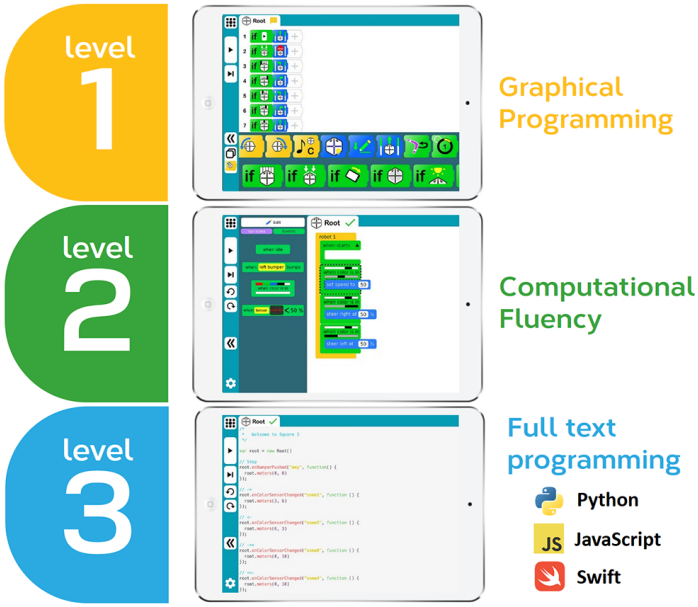


Figura 2.3: Modos de control de Root.

2.1.3. Mirobot

Mirobot [19] es un proyecto comercial de “Mime Industries” para enseñar a usuarios, con distintos niveles de conocimiento, a programar. Se vende el kit de ensamblado, compuesto de un robot desmontado y todos los accesorios necesarios, por £60 (alrededor de 70€). Mirobot dispone de distintos sensores y actuadores: módulo sigue líneas, detección de colisión, es capaz de hacer ruido (bip), y puede moverse por una superficie plana, ejecutando el programa introducido por el usuario en su interfaz web, e ir dejando un rastro pintado con rotulador en dicha superficie. Dispone de una página web [20] alojada en servidores externos al robot, por lo que requiere de una infraestructura o un equipo adicional por parte de los fabricantes de Mirobot para el correcto funcionamiento. Por otro lado, en caso de fallo de dicho servidor, la utilización del robot no sería posible. Asimismo, es necesario que el dispositivo desde el que queramos controlar el robot esté conectado a Internet.

Mirobot dispone además de la posibilidad de utilizar una versión virtual del robot en su interfaz web, de forma que el receptor y encargado de responder a las órdenes no es el robot físico, sino una flecha en la propia interfaz. Naturalmente, esta versión virtual no dispone de sensores.

Figura 2.4: Mirobot.

Mirobot permite controlar el robot desde la interfaz web de varias formas. Podemos programarlo en varios lenguajes: Python 2, JavaScript, Scratch... O podemos controlarlo como si de un vehículo radio control se tratase.

2.2. Proyectos abiertos

2.2.1. TurtleRobot

TurtleRobot [21] es un proyecto FOSS (Free and Open Source Software), publicado en una web de intercambio de proyectos, *Instructables*. Su creador, MakersBox, inició el proyecto con el fin de hacer un taller de 10 horas para ChickTech.org, una organización cuyo objetivo es atraer y retener mujeres jóvenes al campo de la tecnología, y las carreras científico-tecnológicas. El autor se inspiró en Mirobot, y decidió hacer un robot similar, con un menor coste, para su taller.

TurtleRobot está diseñado para ser programado a bajo nivel, programando directamente el robot con la plataforma y el editor de Arduino. Una vez programado, se instala el software en el robot, y éste lo ejecuta. El programa escrito por el usuario debe hacer uso de una librería desarrollada por el autor, para controlar el hardware del robot.

2.2.2. LogoTurtle

LogoTurtle [22] está basado en TurtleRobot, y el hardware es prácticamente el mismo. El creador de este proyecto quiso modernizar LOGO con un hardware nuevo, manteniendo el lenguaje de programación. En la web del proyecto podemos encontrar enlaces a aplicaciones propias a este proyecto, desarrolladas para instalar el programa en el robot. En los enlaces podemos encontrar un compilador de lenguaje LOGO, que lee el código de un fichero de texto plano e instala el programa en el robot.

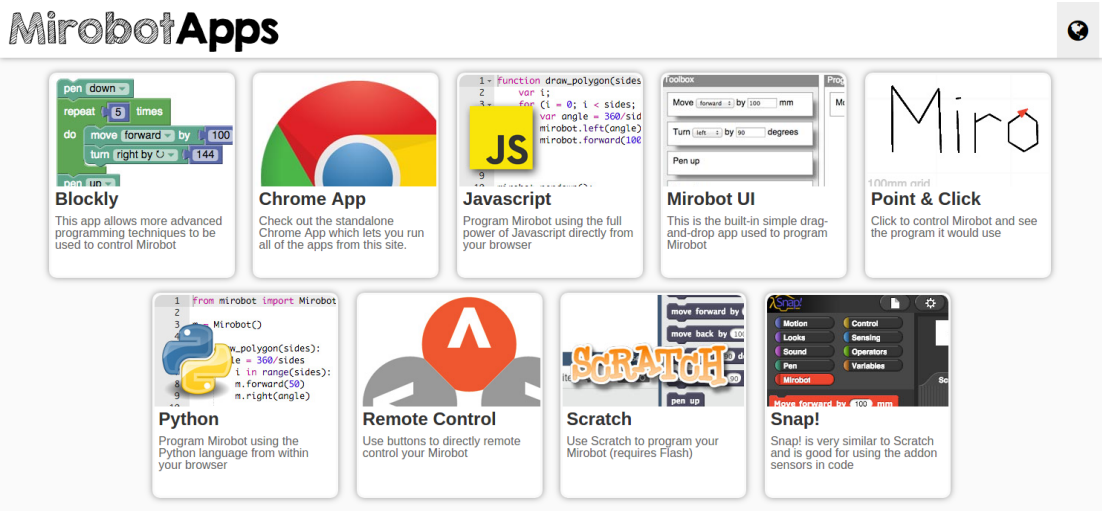


Figura 2.5: Mirobot Apps: Distintas posibilidades para el control de Mirobot.

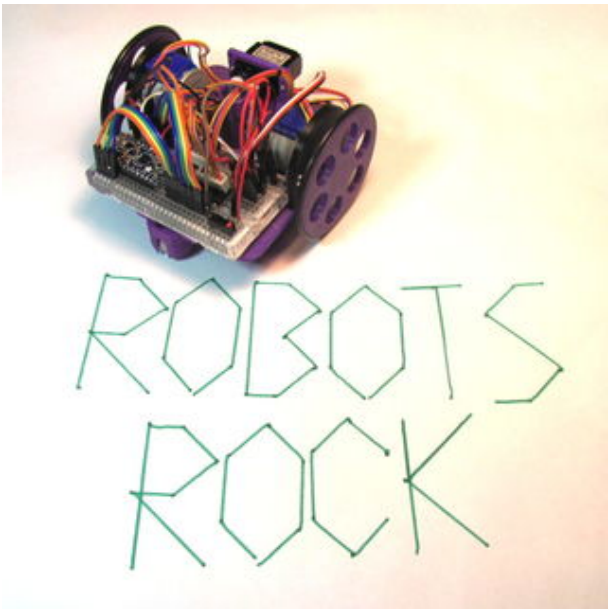


Figura 2.6: TurtleRobot junto a una de sus creaciones.

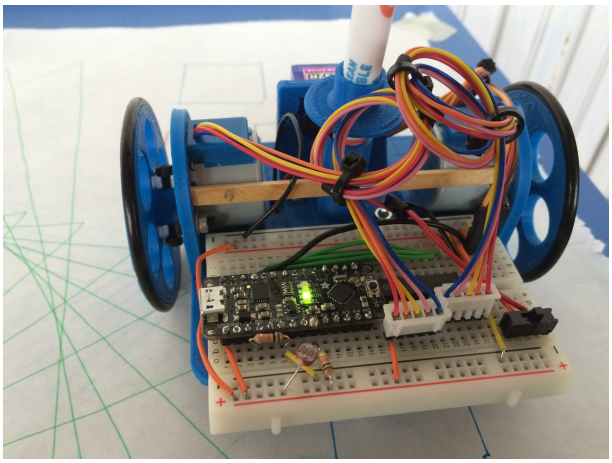


Figura 2.7: LogoTurtle.

2.2.3. Pollock

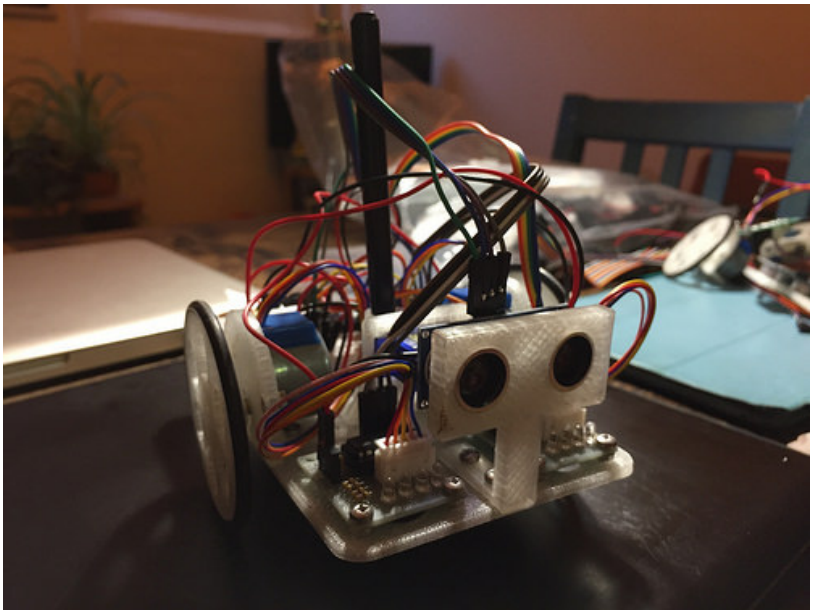


Figura 2.8: Pollock.

Pollock [23] es un robot pintor y explorador creador por el “Phoenix Code Club”. Es un proyecto completamente abierto, y construirlo cuesta menos de £20 (alrededor de 22€). Pollock se basa en TurtleRobot. Las ruedas y el soporte para el rodamiento son los mismos, pero tuvieron que construir un chasis más grande para que cupieran las placas controladoras que muchas veces vienen con los motores paso-paso baratos, así como para que cupieran sensores adicionales, como el sensor de distancia por ultrasonidos que se ve en la figura 2.8.

3

ANÁLISIS

En esta sección se detallan las funcionalidades que debe tener el sistema, a nivel de hardware, firmware[1], software de control e interfaz web de programación.

3.1. Hardware

El hardware es la parte visible y manipulable del robot, por lo que debe estar concebido para que sea funcional, atractivo visualmente, y de un tamaño adecuado para que pueda ser transportado y utilizado sin necesidad de disponer de grandes espacios libres. Debemos poder montar todos los componentes en el chasis y tiene que ser fácil de montar. Dado que está pensado para ser utilizado en superficies planas como mesas, corre el riesgo de caerse por el borde, por lo que debemos hacerlo lo bastante modular para que las piezas dañadas sean fácilmente imprimibles y reemplazables. Para conseguir esto, se diseñará un chasis que pueda ser impreso en una impresora 3D de gama media, con un tamaño de pieza que no sobrepase de las medidas de la plataforma de impresión. El hecho de que sea imprimible en 3D añade la posibilidad de personalización, ya que pueden usarse diferentes colores de filamento para imprimir las distintas partes.

A la hora de seleccionar los componentes para la segunda versión de *Phogo*, el objeto de este Trabajo de Fin de Grado, se deben tener en cuenta los siguientes puntos:

- **Procesamiento**
  La placa utilizada en la primera versión no permitía conexiones WiFi, por lo que se de bería sustituir por otra que sí las permita, y que tenga espacio de almacenamiento suficiente para poder almacenar la interfaz web. De esta forma, el propio robot sería el servidor web y contendría todo el entorno de programación, y a la vez sería el actor del programa tecleado por el usuario.
- **Desplazamiento**
  El movimiento del robot debería ser preciso, o por lo menos consistente. Dado que el objetivo de la mayoría de los programas será que el robot llegue a hacer un dibujo, sería necesaria cierta precisión en el desplazamiento, puesto que si no, por ejemplo, no se podrían dibujar figuras cerradas.
- **Sensores**
  Se quiere que el robot tenga la posibilidad de captar datos del mundo real, y que estos datos puedan ser incorporados a los programas. Por ello, se le debería dotar de un sensor simple, como un sensor de distancia por ultrasonidos. La incorporación de este sensor permitiría incorporar cierta lógica en

[1] El firmware es el programa que establece la lógica de más bajo nivel y que controla el hardware de ciertos dispositivos.

los programas y nos permitiría introducir conceptos de programación como los flujos condicionales. Sería relativamente trivial programar nuestro robot para que sea capaz de resolver laberintos [24].

- **Actuadores**
  Se dotará al de un rotulador para que vaya marcando el suelo por donde pasa. Esta función debería ser también controlable por el usuario, por lo que necesitaríamos un servomotor que controlase si el robot va a pintar o no, manipulando el rotulador.

3.2. Software

El software de *Phogo* debería ser ligero, puesto que los procesadores disponibles en robótica no suelen destacar por su potencia de procesamiento y no tienen la misma potencia que un procesador de ordenador al uso.

Protocolo de intercambio de datos

El protocolo de intercambio de datos entre la interfaz y el robot tiene que ir sobre el protocolo HTTP, que es el utilizado en la web. Debe ser un protocolo extensible, conciso y ligero, pero lo bastante simple y declarativo para que se entienda sin conocer los detalles de implementación.

3.3. Web

Por su sencillez de puesta en marcha y la familiaridad con la que casi todo el mundo maneja los navegadores web, y las aplicaciones de navegador, la interfaz de control para *Phogo* se va a desarrollar en forma de página web, puesto que no requiere ningún esfuerzo de configuración por parte del usuario, ni ninguna infraestructura o software adicional.

Se concibe la interfaz como un entorno de desarrollo amigable y sencillo de usar. Debería incluir un editor de texto en el que se pudiera introducir el código que debe ejecutar el robot, así como un elemento que permitiera mostrar la respuesta del robot una vez haya ejecutado los comandos. Dado que hoy en día existen muchos dispositivos distintos capaces de conectarse a Internet y cada uno tiene unas características distintas, y muy diferentes entre sí, se debería hacer la interfaz responsive[2].

[2]El diseño web adaptable o *responsive* es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visitarlas. Esta tecnología pretende que con un único diseño web, todo se vea correctamente en cualquier dispositivo.

4

Desarrollo

4.1. Hardware

El chasis que forma la estructura de *Phogo* es modular e imprimible en 3D. Las distintas partes se ensamblan por medio de tornillos y tuercas, bridas, o cualquier otro elemento que permita fijarlas, siempre que las diferentes partes no queden fijas permanentemente las unas a las otras, para que se puedan reemplazar en caso de necesidad.

Los 4 puntos a tener en cuenta a la hora de seleccionar los componentes electrónicos del robot son los siguientes:

- **Procesamiento**
  La placa bq Zum utilizada en la primera versión, con un coste aproximado de 40€, no cumple los requisitos necesarios para usarla en esta versión. Sustituyéndola por el chip ESP8266 integrado en la placa NodeMCU, con un coste aproximado de 6€, mantenemos el entorno de desarrollo Arduino, pudiendo establecer conexiones WiFi, a una fracción del precio. Podemos utilizar entonces las capacidades de este chip para proporcionar un servidor web, y usar una página web como interfaz de programación.

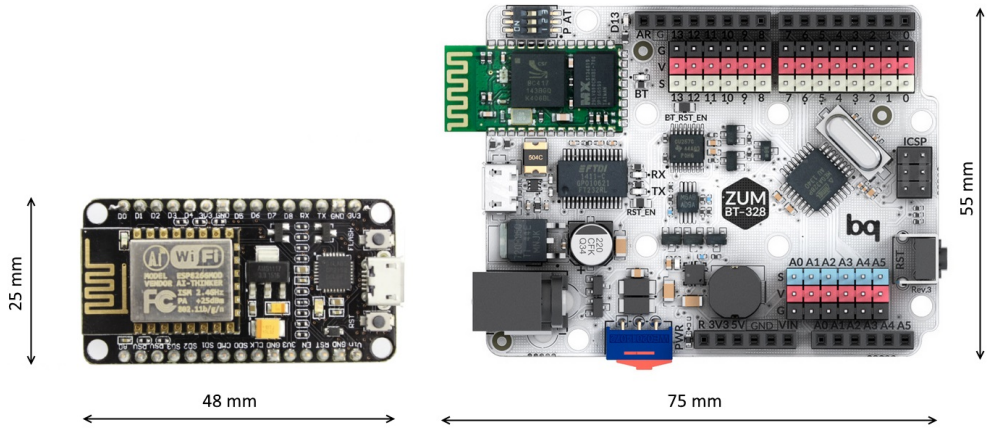


Figura 4.1: NodeMCU a la izquierda y bq Zum a la derecha.

- **Movimiento**
  Para que la funcionalidad de la que queremos dotar al robot sea posible, necesitamos motores paso-paso, para que tenga una mínima precisión. Este tipo de motores necesita una placa controladora,

que es la encargada de decirle cómo tiene que moverse. Los motores seleccionados son los motores 28BYJ-48, muy utilizados en robótica por su pequeño tamaño y su bajo precio, proporcionando una calidad de movimiento aceptable.

Figura 4.2: Motor 28BYJ-48 y driver ULN2003.

- **Sensores**

  Para poder usar datos del mundo real en nuestro programa, necesitamos sensores. En este caso, se va a incorporar un sensor de distancia por ultrasonidos. El sensor elegido es el sensor HC-SR04, que nos proporciona una medida de un rango de distancias adecuado a nuestra escala (entre 5 y 500 cm).

Figura 4.3: Sensor de distancia HC-SR04.

- **Actuadores**

  Servomotor para el control del rotulador. Este servomotor no va soportar una gran carga, ni necesitamos que sea especialmente rápido. Lo que sí es necesario, es que sea de tamaño reducido. El servo motor utilizado es el microservo *Tower Pro 9g*.

La lista de materiales se encuentra detallada en el Anexo B.

Figura 4.4: Microservo Tower Pro 9g.

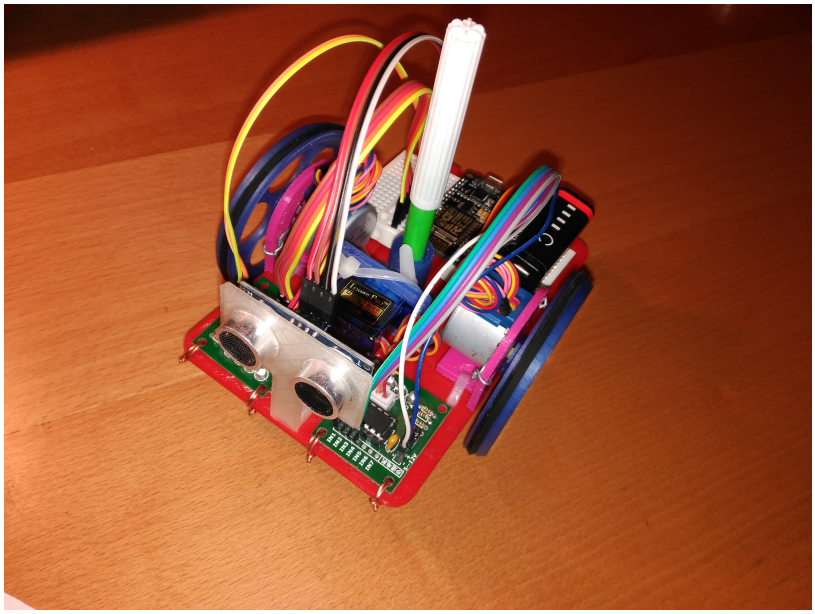
4.2. Software

Para poder distribuir *Phogo* de forma abierta y gratuita en el repositorio del proyecto [25], todo el software y librerías de terceros que se van a utilizar está licenciado de forma que se permite la libre modificación y distribución, por lo que no se incurrirá en violaciones de licencias al utilizarlo como parte de este proyecto y distribuirlo.

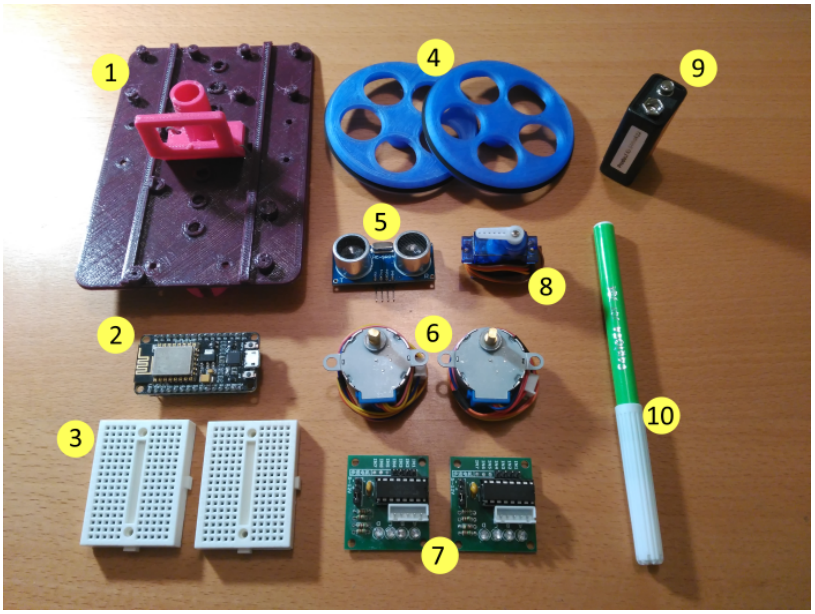
El chip ESP8266 que se utiliza va integrado en la placa de desarrollo NodeMCU, que provee funcionalidades adicionales a las del propio chip, tales como una interfaz USB para programarlo, GPIOs (General Purpose Input Output) en formato protoboard, USB para alimentación de la placa... La placa NodeMCU fue diseñada originalmente para ejecutar código Lua, pero la comunidad de usuarios desarrolló controladores para adaptarla al entorno de desarrollo Arduino, haciéndola compatible con dicha plataforma. Tiene un precio aproximado de 6€.

El desarrollo del firmware de *Phogo* se ha realizado con el entorno de programación Arduino, usando las librerías de código propias del ESP8266, además de las siguientes librerías adicionales, para dotarle de toda la funcionalidad deseada:

- **ESP8266WebServer**
  Permite utilizar el ESP8266 como servidor web. Con esta librería sólo se puede dar servicio a un cliente a la vez, lo que es perfecto para nuestro caso de uso.
- **ESP8266mDNS**
  Permite publicar en la red local como servicio Bonjour/Avahi con una dirección conocida (phogo.local) y válida sólo dentro de la red local. De esta forma el navegador que usemos como cliente podrá encontrar el servidor sin necesidad de conocer la dirección IP del servidor, ya que esta puede cambiar según la red WiFi a la que esté conectado.
- **Servo**
  Facilita el uso del servo motor.
- **ArduinoJson**
  Parseador y constructor de lenguaje JSON. Se utiliza para leer y construir los mensajes entre el navegador y el robot.



(a) Una Tortuga (versión 2) completa.



(b) Las partes de la Tortuga: 1) Chasis imprimible en 3D, 2) Placa de control: NodeMCU, 3) Placas de prototipado tamaño mini, 4) Ruedas imprimibles en 3D, 5) Sensor de distancia, 6) Motores paso-paso, 7) Placas de control de los motores, 8) Microservo para controlar el rotulador, 9) Batería de 9V, 10) Rotulador

Figura 4.5: La segunda versión de *Phogo*.

- **WiFiClient**
  Permite la conexión a redes WiFi ya existentes.
- **ESP8266WiFi + ESP8266WiFiMulti**
  Permite crear un punto de acceso WiFi.
- **AccelStepper**
  Facilita el manejo de los motores paso-paso, y les dota de parámetros como la aceleración, velocidad máxima, etc. si lo necesitamos.

La creación de un Punto de Acceso (AP) WiFi o conexión a uno existente es una pieza clave en el correcto funcionamiento del sistema y debe realizarse en primer lugar, al arranque del robot. El flujo que se desea implementar es el mostrado en la figura 4.6. Sin embargo, el almacenaje y manejo de varios puntos de acceso no forma parte de los objetivos principales de este Trabajo de Fin de Grado, y se considerará su implementación en función del estado de los demás objetivos.

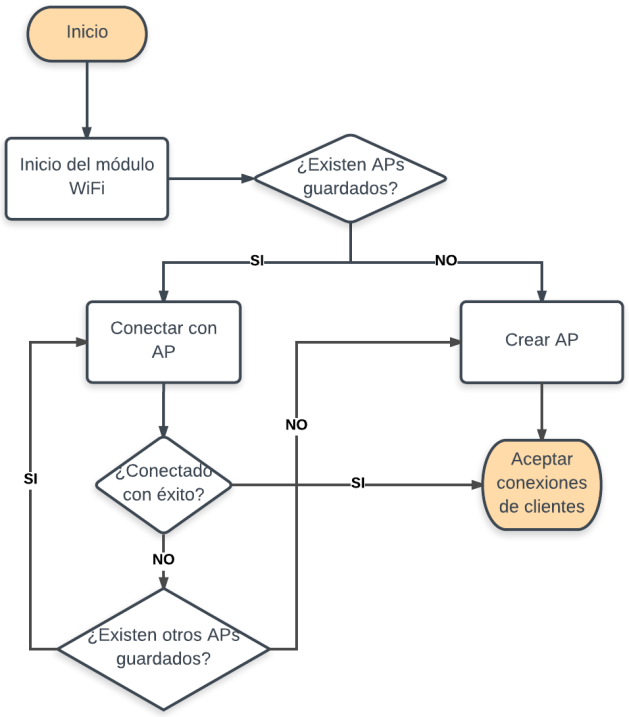


Figura 4.6: Esquema del flujo de creación o conexión a un Punto de Acceso WiFi.

4.3. Web

Para el desarrollo de la web, se han utilizado herramientas abiertas. Para el manejo del DOM[1], se ha usado jQuery [26] (necesario también para el uso de la terminal embebida y el editor de código). Para hacer la web responsive, se ha usado la librería Bootstrap 3 [27].

[1] *Document Object Model* es una interfaz de programación (API) para HTML válido y documentos XML bien formados. Define la estructura lógica de los documentos y la forma en que se accede a ellos y se manipulan.

- **Editor de código**

  El editor web para código más popular es ACE [28]. Nos permite personalizarlo con temas, resaltado de sintaxis para multitud de lenguajes, usar snippets de código en muchos lenguajes, autocompletado... Es muy modular, y nos permite utilizar sólo las funcionalidades que necesitemos, por lo que no es necesario que esté todo el proyecto disponible para que funcione el editor. Vamos a utilizar únicamente los módulos `mode-python3.js`, `theme-monokai.js` y la extensión para formateo de código `ext-beautify.js`.

- **Intérprete de Python 3**

  El intérprete de Python 3 utilizado es Brython (**BR**owser p**YTHON**). Brython es un transpilador[2] de Python 3 a JavaScript, que es el único lenguaje que puede ejecutar el navegador. Es un proyecto actualmente en desarrollo con varias versiones estables. Brython permite, en tiempo de desarrollo, listar los paquetes Python que se han utilizado en el proyecto y generar un archivo que contiene únicamente dichos paquetes, de forma que no tenemos por qué incluir toda la librería estándar de Python en el proyecto [29]. Esta optimización reduce mucho la sobrecarga no productiva en la red, el almacenamiento necesario, y el procesado de la web. Tendremos únicamente los paquetes necesarios para ejecutar nuestra aplicación, y en un sólo archivo en vez de en varios. Con esto conseguimos descargar en el cliente todo el código con una sola petición HTTP.

- **Terminal embebida**

  jQuery Terminal nos permite programar distintos intérpretes de comandos en una terminal que podemos embeber en la web [30], proporcionando una experiencia similar a una terminal de Linux. Actualmente usamos jQuery Terminal únicamente como salida (stdout y stderr), para mostrar en la interfaz las respuestas del robot. Se han redirigido las salidas estándar de Brython, `sys.stdout` y `sys.stderr` a dos clases Python que utilizan la terminal embebida, de forma que al ejecutar la función `print(...)` y `print(..., file=sys.stderr)`, en vez de escribir en la consola de las herramientas de desarrollo del navegador, como hace por defecto, la salida se hace por la terminal, en color rojo si escribimos en stderr.

Herramientas y flujo de desarrollo

Durante la fase de desarrollo de este Trabajo de Fin de Grado, para evitar muchos de los inconvenientes que tiene trabajar con un robot físico, como por ejemplo el tiempo de carga de los archivos al almacenamiento interno del robot, se ha desarrollado un pequeño servidor web que imita las respuestas del robot (escrito en Python, usando el microframework Flask [31]). Este servidor de desarrollo se comunica también con la interfaz de control para determinar qué módulos Python de la librería estándar se han importado (y por tanto utilizado, directa o indirectamente por algún otro módulo) y empaqueta los módulos Python necesarios para ejecutar Brython con lo mínimo necesario para nuestro proyecto.

Para agilizar un poco más el desarrollo de la interfaz, se han utilizado herramientas de desarrollo de NodeJS con las que se ha definido un proceso de adecuación de los archivos JavaScript y CSS. Recordemos que el ESP8266 puede almacenar hasta 4MB de archivos, por lo que los archivos fuente deben tratarse para reducir su tamaño. Este tratamiento consiste en eliminar todos aquéllos espacios y caracteres en blanco que no sean estrictamente necesarios para el funcionamiento del código, así como el renombrado de las variables, además de otras transformaciones, con el objetivo de reducir el número total de caracteres, y con ello reducir el tamaño

[2] Software que convierte el código fuente de un lenguaje de programación en código fuente de otro lenguaje de programación. Por ejemplo, CoffeeScript, Caffeine, Kaffeine y más de otra docena de lenguajes son transpilados a JavaScript.

final del archivo. Este proceso se conoce como “minificación” [32].

Este chip ESP8266 dispone de un sistema de archivos de 4MB para almacenamiento de datos (aparte del código del robot). Por lo tanto tenemos espacio para almacenar la web, aunque un espacio bastante reducido para almacenar toda la interfaz de control, librerías adicionales necesarias. Como referencia, diremos que una foto hecha con un móvil moderno, con una resolución de 12 megapíxeles, ocupa alrededor de 3.5MB. Sin olvidar que el ESP8266 tiene que ser capaz de servir la web en un tiempo razonable, y no dispone de múltiples hilos de ejecución, como los servidores tradicionales, para enviar los distintos archivos de los que se compondrá la web de forma paralela. Se debe hacer una interfaz sencilla, funcional, y ligera, compuesta del menor número de archivos posible. No tiene intérprete de Python 3 incorporado, ni es posible instalarlo puesto que no dispone de sistema operativo, por lo que tenemos que ejecutar (interpretar) el código en el navegador, y comunicar al robot las acciones que tiene que realizar por dicha ejecución. Para el intercambio de datos entre el navegador y el robot, se ha definido un formato de intercambio de datos, detallado en el Anexo A.

5

Taller de iniciación a la programación con *Phogo*

Para la realización del taller con alumnos, y posterior evaluación de lo aprendido, se ha diseñado el siguiente experimento:

Los alumnos recibirán una pequeña documentación sobre el lenguaje, y qué esperar cuando empiecen a escribir código. Esta documentación contendrá pequeños ejemplos de código, convenciones sobre los nombres de las variables y las funciones, sobre la forma de escribir código Python (indentación, puntuación...) y de los errores más comunes al utilizar Python por primera vez, como por ejemplo `SyntaxError`, `IndentationError`, `ImportError`, y una breve descripción de cuales podrían ser los motivos por los que ocurren estos errores. Es decir, un pequeño resumen del lenguaje.

Para aprovechar al máximo las capacidades artísticas de *Phogo*, se llevarán a cabo una serie de retos en los que los participantes tendrán que programar su *Phogo* para que copie una serie de dibujos, cada uno de los cuales introducirá algún concepto nuevo, ampliando los anteriores.

Los alumnos que participen recibirán instrucciones para cada uno de los retos que se les planteen, explicando el concepto de programación a introducir, y proponiendo un reto nuevo, para el que deberán aplicar el nuevo concepto, así como los anteriores. Cada reto tendrá un tiempo máximo de realización, antes de pasar al siguiente, y se les pedirá que cada uno guarde el código que resuelva cada reto. Posteriormente, se podrá analizar el código para comprobar si cumple o no con las condiciones establecidas.

6

Pruebas y evaluación de resultados

Toda la funcionalidad relativa al robot se ha implementado correctamente:

- El robot se conecta un Punto de Acceso WiFi si está dentro del alcance del único Punto de Acceso que podemos configurar. De lo contrario, crea uno propio.
- El robot inicia un servidor web, que puede dar servicio a un solo cliente.
- El robot almacena la interfaz de control y es capaz de servirla cuando se conecta un cliente.
- El robot ejecuta las acciones definidas y solicitadas por el cliente, devolviendo el resultado de la ejecución.

En la interfaz web se han implementado todas la funcionalidades previstas, además de algunos extras:

- Tenemos un editor de código Python 3, con resaltado de sintaxis, formateo de código y personalizable.
- Tenemos una terminal embebida en la web, que sirve como salida de la ejecución de los comandos en el robot, y además podemos programarla para que tenga cierta funcionalidad, como introducción a la terminal de Linux.
- Podemos guardar en nuestro ordenador el código escrito en el editor como archivo Python, y podemos restaurarlo posteriormente.
- Se ha diseñado e implementado un protocolo de intercambio de datos simple y conciso, fácilmente ampliable si se quiere dotar al sistema de más funcionalidad en el futuro.

Pese a que el diseño del experimento se considera satisfactorio y se piensa que daría resultados en favor de las plataformas educativas alternativas como *Phogo* basándonos en el taller que sí se llevó a cabo con la primera versión, el taller de iniciación a la programación con *Phogo* no pudo llevarse a cabo: es difícil encontrar un grupo de estudiantes que se presten a ello.

Figura 6.1: Interfaz web. A la izquierda, el editor de código. A la derecha, la terminal embebida. En la terminal se muestra el resultado de la ejecución del código del editor.

7

Conclusiones, Problemas y Trabajo futuro

En este apartado se exponen las conclusiones obtenidas tras el desarrollo del proyecto, seguidas de los fallos conocidos del sistema, y finalmente se detallan algunas líneas de trabajo futuro.

7.1. Conclusiones

Tras el desarrollo de este Trabajo de Fin de Grado, se ha obtenido un robot programable en lenguaje Python vía navegador web. El robot es capaz de ejecutar una serie de acciones básicas pero con el potencial de ser combinadas de incontables formas para crear dibujos complejos. Dado que hace las veces de servidor web y la interfaz se comunica con él, tenemos una herramienta autocontenida, que no necesita de configuraciones complejas o software especial, ni despliegues en servidores externos, facilitando el uso en entornos domésticos o aulas.

El desarrollo de *Phogo* ha sido más complicado de lo que se esperaba en un principio puesto que el proyecto involucra muchas tecnologías y lenguajes de programación diferentes, a muchos niveles.

Phogo se ha rehecho desde cero, manteniendo únicamente las funciones públicas disponibles para manejarlo. Se ha rediseñado el chasis del robot, revisado la selección de componentes electrónicos y su integración en el robot, cambiando todo aquéllo que no dio buen resultado en la primera versión. Se ha hecho también el software que mueve al robot, escrito en Arduino (C/C++), la interfaz web, escrita en HTML5, JavaScript y CSS3. El control de *Phogo* en la web está escrito en Python 3, que se procesa con Brython, el intérprete y transpilador de Python 3, e interacciona con JavaScript a través de código propio, por ejemplo, para poder utilizar la terminal embebida desde Python, la gestión de las comunicaciones, etc.

Todo ello, además, con las limitaciones que nos imponía el hardware (4MB de almacenaje de datos), y aquéllas marcadas por los objetivos principales del proyecto: coste reducido y facilidad de montaje. Tras varias iteraciones, optimizaciones de código y de procesos, el tamaño final de la interfaz web es 3.7MB, y el coste de *Phogo*, excluyendo el material de impresión 3D, se ha mantenido por debajo de los 21€, reduciendo el coste un 75 % con respecto a la primera versión.

7.2. Problemas conocidos

Los problemas que se conocen tienen que ver más con el entorno de ejecución de los programas de usuario que con el sistema. Es decir, al ejecutar el código en el navegador, entran en juego otros factores aparte del

propio código, como por ejemplo, qué navegador utilicemos.

En Google Chrome, la interfaz se queda bloqueada durante la ejecución del código. Debido a la naturaleza asíncrona de JavaScript, que contrasta con el sincronismo de Python, se han tenido que hacer síncronas las peticiones al servidor. De lo contrario, no habría sido posible ejecutar las acciones en el orden en que se escriben. Este sincronismo de las peticiones al servidor, y puesto que JavaScript se ejecuta en un solo hilo, hacen que la interfaz se congele durante la ejecución. Existe sin embargo la posibilidad de programar en pseudo multi-hilo usando los "Workers" del navegador [33], pero Brython no puede ejecutarse por entero en un "worker" porque tiene que acceder al DOM, y desde los "workers" no está permitido.

En Mozilla Firefox, sin embargo, no se bloquea la interfaz, aunque se desconoce el motivo. Mientras se investiga la forma de solucionar esta congelación de la web, Mozilla Firefox será el navegador recomendado para el uso de *Phogo*.

7.3. Trabajo futuro

Los objetivos del Trabajo de Fin de Grado se consideran cumplidos con éxito. Existen sin embargo ciertas líneas de trabajo que permitirían mejorar el proyecto en un futuro. Se piensa continuar con el desarrollo de este proyecto mejorando la experiencia de usuario en el uso de la interfaz web, mejorando los tiempos de respuesta y de carga.

Se quiere añadir al sistema una interfaz de administración de puntos de acceso WiFi, con el fin de poder añadir, eliminar y modificar los puntos de acceso almacenados en el robot.

Otro añadido sería una versión virtual de la Tortuga, similar a la existente en la primera versión de *Phogo*, pero contenida en la interfaz web.

Se planea implementar una serie de comandos de control de la interfaz web ejecutables desde la terminal embebida (por ejemplo poder ejecutar el código sin pulsar el botón "Run'), con el objetivo de acercar un poco más al usuario a un entorno de programación por terminal, dónde no sólo se ve el resultado de la ejecución, sino que se pueden introducir comandos. Se plantea la posibilidad de incluir asimismo un intérprete de Python 3 interactivo.

En futuras versiones, se plantea la posibilidad de cambiar el chip ESP8266 por su sucesor, el ESP32, con más capacidad de procesamiento, memoria y GPIOs, aunque un poco más caro (entre 15€ y 50€, dependiendo de la placa de desarrollo en que se encuentre integrado), con el fin de incluir una interfaz más completa, aunque posiblemente más pesada y ampliar la gama de sensores y actuadores disponibles en el robot.

Por falta de oportunidad, no se ha dado a probar esta versión de *Phogo* a alumnos reales. Se espera poder realizar en el futuro el taller planteado.

Bibliografía

[1] E. Sadin, *La vie algorithmique. Critique de la raison numérique*. L'échappée, 2015.

[2] L. Manovich, *Software takes command*. Bloomsbury, 2014.

[3] D. Rushkoff, *Program or Be Programmed. Ten Commands for a Digital Age*. OR Books, 2010.

[4] I. Paliokas, C. Arapidis, and M. Mpimpitsos, *Game Based Early Programming Education: The More You Play, the More You Learn*, pp. 115–131. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[5] J.-M. Sáez-López, M. Román-González, and E. Vázquez-Cano, "Visual programming languages integrated across the curriculum in elementary school: A two year case study using "scratch" in five schools," *Computers & Education*, vol. 97, pp. 129 – 141, 2016.

[6] A. Gomes and F. Brito Correia, "Programming education strategies," in *ICERI2014 Proceedings*, 7th International Conference of Education, Research and Innovation, pp. 2551–2560, IATED, 17-19 November, 2014 2014.

[7] J. Moreno-León, G. Robles, and M. Román-González, "ode to learn: Where does it belong in the k-12 curriculum?," *Journal of Information Technology Education: Research*, vol. 15, pp. 283–303, 2016.

[8] F. B. V. Benitti, "Exploring the educational potential of robotics in schools: A systematic review," *Computers & Education*, vol. 58, no. 3, pp. 978 – 988, 2012.

[9] R. Burbaitė, R. Damaševičius, and V. Štuikys, "Using robots as learning objects for teaching computer science," in *X world conference on computers in education*, pp. 101–110, 2013.

[10] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, *et al.*, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[11] C. García-Saura and J. González-Gómez, "Low cost educational platform for robotics, using open-source 3d printers and open-source hardware," in *ICERI2012 Proceedings*, pp. 2699–2706, IATED, 2012.

[12] S. A. Papert, "Teaching children thinking," Artificial Intelligence Memos LOGO Memo No. 2, Massachusetts Institute of Technology - A. I. Laboratory, October 1971.

[13] L. Mannila and M. de Raadt, "An objective comparison of languages for teaching introductory programming," in *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006*, Baltic Sea '06, (New York, NY, USA), pp. 32–37, ACM, 2006.

[14] C. Gonzalez-Sacristan, C. Garcia-Saura, and P. Molins-Ruano, "Phogo: A low cost, engaging and modern proposal to learn how to program," in *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*, pp. 67–71, ACM, 2016.

[15] C. García-Saura, "Self-calibration of a differential wheeled robot using only a gyroscope and a distance sensor," Master's thesis, Imperial College London, 2015.

[16] "Lego Mindstorms." `https://www.lego.com/en-us/mindstorms`https://www.lego.com/en-us/mindstorms. Último acceso: 05/05/2017.

[17] "Official Website of Root | The robot that brings code to life." `http://www.codewithroot.com/`. Último acceso: 05/05/2017.

[18] "Root - A robot to teach coding by Scansorial — Kickstarter." `https://www.kickstarter.com/projects/1509453982/root-a-robot-to-teach-coding`. Último acceso: 05/05/2017.

[19] "Mime Industries - Technology kits for kids." https://mime.co.uk/. Último acceso: 05/05/2017.

[20] "Mirobot Apps." http://apps.mirobot.io. Último acceso: 05/05/2017.

[21] "Low-Cost, Arduino-Compatible Drawing Robot." http://www.instructables.com/id/Low-Cost-Arduino-Compatible-Drawing-Robot/. Último acceso: 05/05/2017.

[22] "LogoTurtle." http://archive.monograph.io/joshburker/logoturtle. Último acceso: 05/05/2017.

[23] "Interact Digital Arts - Pollock - Autonomous Drawing Robot." http://interactdigitalarts.uk/pollock. Último acceso: 05/05/2017.

[24] S. A. Papert, "Uses of technology to enhance education," Tech. Rep. LOGO Memo No. 8, Massachusetts Institute of Technology - A. I. Laboratory, June 1973.

[25] C. González, "Phogo v2." https://github.com/ChacheGS/Phogo_v2. Último acceso: 26/06/2017.

[26] "jQuery." https://jquery.com/. Último acceso: 05/05/2017.

[27] "Bootstrap." http://getbootstrap.com/. Último acceso: 15/04/2017.

[28] "Ace - The High Performance Code Editor for the Web." https://ace.c9.io/. Último acceso: 10/02/2017.

[29] "Brython Documentation." http://www.brython.info/static_doc/en/intro.html?lang=en. Último acceso: 01/06/2017.

[30] J. Cubic, "jQuery Terminal Emulator Plugin - API Reference." http://terminal.jcubic.pl/api_reference.php. Último acceso: 02/06/2017.

[31] "Flask (A Python Microframework)." http://flask.pocoo.org/. Último acceso: 03/12/2016.

[32] D. Crockford, "JSMin The JavaScript Minifier." http://www.crockford.com/javascript/jsmin.html, 2003. Último acceso: 27/03/2017.

[33] Mozilla Development Network, "Using Web Workers." https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers. Último acceso: 20/05/2017.

I

Anexos

A

Protocolo de intercambio de datos

El protocolo definido para el intercambio de datos entre el navegador y el robot es muy sencillo: cada comando tiene un número identificador, siempre incremental. Este identificador se utiliza como contador de mensajes, y la correlación incremental, para comprobar que las instrucciones se han ejecutado en el orden deseado.

Petición

Se definen dos campos, el campo “action” para la acción a realizar por el robot (obligatorio), y otro campo “params” que es opcional y almacena tantos parámetros como sean necesarios para ejecutar la acción. Si el comando no requiere parámetros, el campo “params” no se incluye.

```
{
    "id": "",
    "cmd": {
        "action": "",
        "params": {
            "param1": "",
            "param2": ""
            // etc
        }
    }
}
```

Respuesta

La respuesta contiene la misma información que la petición, con un nuevo campo “result”, que es obligatorio e indica el resultado de la ejecución de la acción.

```
{
    "id": request["id"],
    "cmd": {
        "action": request["cmd"]["action"],
        "params": request["cmd"]["params"]
    },
```

```
    "result": ""
}
```

Ejemplo un comando sin parámetros: distance()

Petición

```
    {
        "id": 0,
        "cmd": {
            "action": "distance"
        }
    }
```

Respuesta

```
    {
        "id": 0,
        "cmd": {
            "action": "distance"
        },
        "result": 50
    }
```

Ejemplo un comando con parámetros: forward(units=40)

Petición

```
    {
        "id": 1,
        "cmd": {
            "action": "forward",
            "params": {
                "units": 40
            }
        }
    }
```

Respuesta

```
    {
        "id": 1,
        "cmd": {
            "action": "forward",
            "params": {
                "units": 40
            }
        },
        "result": "OK"
    }
```

B

Lista de materiales

Se incluye la lista de los componentes electrónicos utilizados para el desarrollo del proyecto, así como enlaces a la tienda donde se adquirieron.

1.– Motores paso-paso:

- link: `http://www.ebay.es/itm/201522335417`
- cantidad: 2

  Precio total: 2 * 2.35€ = 4.70€

2.– NodeMCU ESP8266:

- link: `http://www.ebay.es/itm/201612446249`
- cantidad: 1

  Precio total: 1 * 5.80€ = 5.80€

3.– Micro servo:

- link: `http://www.ebay.es/itm/201528664196`
- cantidad: 1

  Precio total: 1 * 1.88€ = 1.88€

4.– Sensor de distancia por ultrasonidos:

- link: `http://www.ebay.es/itm/201590799594`
- cantidad: 1

  Precio total: 1 * 1.25€ = 1.25€

5.– Mini protoboard:

- link: `http://www.ebay.es/itm/201472319767`
- cantidad: 2

  Precio total: 2 * 1€ = 2€

6.– Cables:

- link: `http://www.ebay.es/itm/201475180578`
- cantidad: 1
- link: `http://www.ebay.es/itm/201530740090`
- cantidad: 1

  Precio total: 1 * 2.25€ + 1 * 2.25€ = 4.5€

Precio total de los componentes: 4.70€ + 5.80€ + 1.88€ + 1.25€ + 2€ + 4.5€ = 20.13€